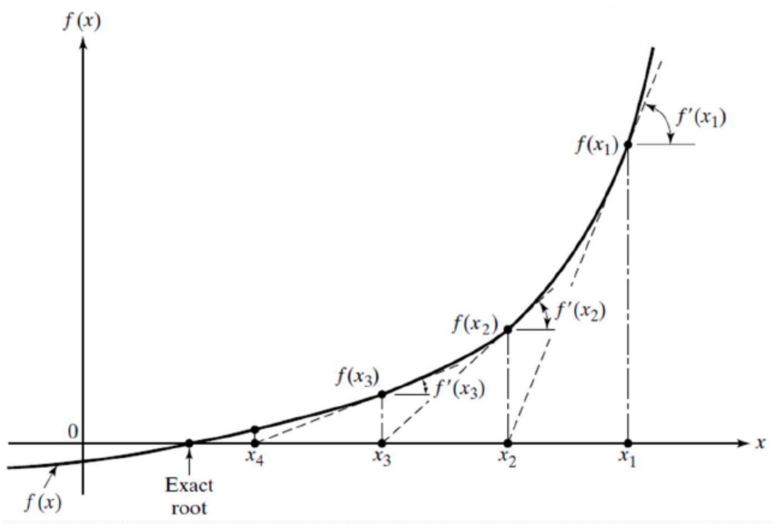
2º SEMESTRE: CÁLCULO NUMÉRICO CON FORTRAN.

TEMARIO:

1. Sistemas lineales. Directos e iterativos.
2. Cálculo de autovalores.
3. Derivación e integración numérica.
4. Solución de ecuaciones y sistemas no lineales.
5. Solución numérica de ecuaciones diferenciales.

Método de Newton-Raphson (1D).



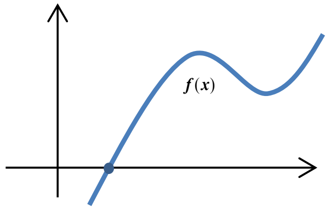
Partimos de un punto inicial x_0

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

i=i+1

$$\text{Si } \left| f(x_{i+1}) \right| < Tol \rightarrow x^* = x_{i+1}$$

RESOLUCIÓN NUMÉRICA DE ECUACIONES NO LINEALES



MÉTODO DE NEWTON: Algoritmo

Tomamos el desarrollo de Taylor de la función

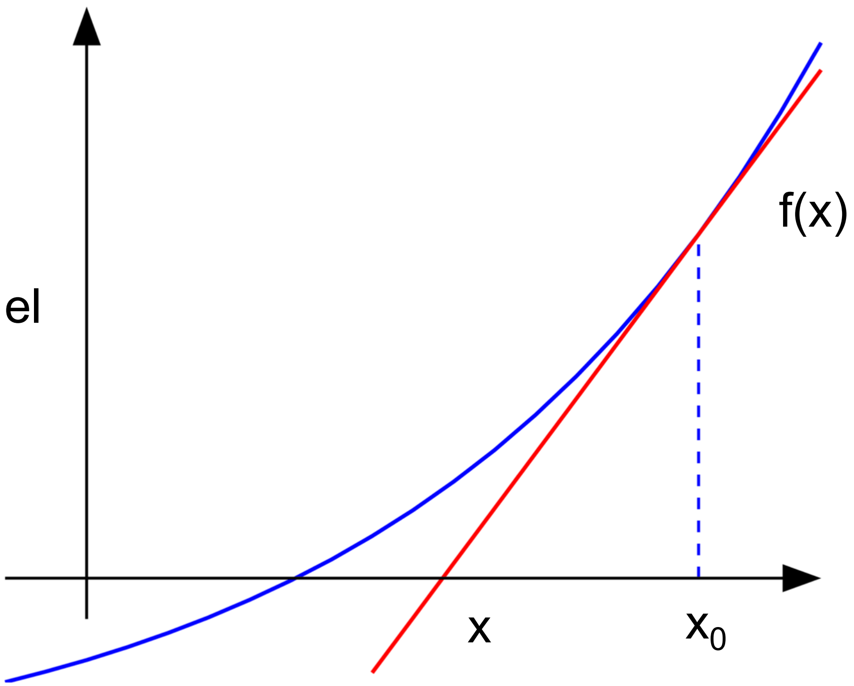
$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \dots$$

Si tomamos solo los términos de primer orden, nos queda la ecuación de la recta tangente en el punto x_0. Para despejar el punto donde la tangente vale cero:

$$0 = f(x_0) + f'(x_0)(x - x_0)$$

Y despejando x

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \longrightarrow x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



Método de Newton-Raphson (1D).

```
subroutine newton(x,f,df,tol,max_iter,unit)

real*8,intent(inout)            :: x
real*8,intent(in)               :: tol
integer,intent(in)              :: max_iter,unit

interface
        function f(x)
                real*8          :: x
                real*8          :: f
        end function

        function df(x)
                real*8          :: x
                real*8          :: df
        end function
end interface

! locales
real*8                          :: x0

x0 = x
do iter = 1, max_iter

        x = x0 - (f(x0))/(df(x0))   ! punto medio del intervalo

        if ((abs((x-x0)/x)<tol).and.(abs(f(x))<tol))  exit  ! control de error

        x0 = x
enddo

end subroutine
```

Método de Newton-Raphson (Sistemas).

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases} \rightarrow F(x_1, x_2, \dots, x_n) = 0$$

Aproximación polinomio de Taylor:

$$F(X^{n+1}) = F(X^n) + J(X^n)(X^{n+1} - X^n) + O(X^{n+1} - X^n)^2$$

Buscamos ceros de la función F:

$$0 = F(X^n) + J(X^n)(X^{n+1} - X^n) \rightarrow \boxed{X^{n+1} = X^n - J^{-1}(X^n)F(X^n)}$$

SISTEMAS NO LINEALES

Método de Newton-Raphson (Sistemas).

$$X^{n+1} = X^n - J^{-1}(X^n)F(X^n) \qquad \text{con } J(X^n) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

$$X^{n+1} = X^n - \underbrace{J^{-1}(X^n)F(X^n)}_{Y^n} \rightarrow \boxed{X^{n+1} = X^n - Y^n}$$

$$J(X^n)Y^n = F(X^n) \qquad (Ax=B)$$

Método de Newton-Raphson (Sistemas).

Solución inicial: X^0

Método en dos pasos: (Bucle)

- Resolver: $J(X^n)Y^n = F(X^n)$
- Evaluar: $X^{n+1} = X^n - Y^n$

- Calcular el error relativo: $\| X^{n+1} - X^n \|$

Complemento:

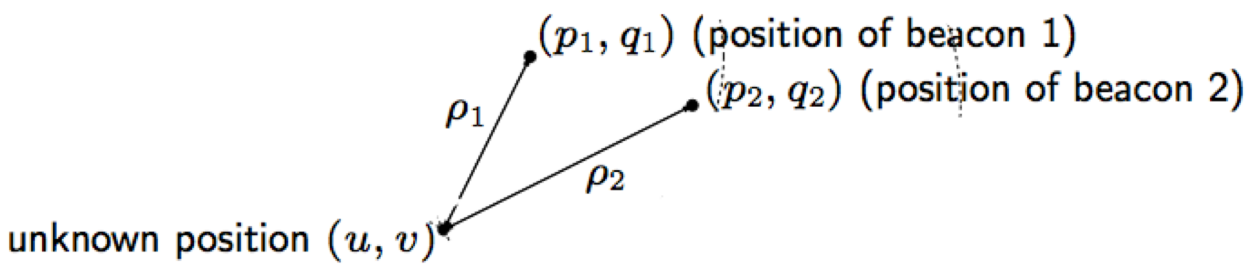
HERRAMIENTA DE NAVEGACIÓN

Datos de entrada:

Posición de baliza 1
Posición de baliza 2
Distancia a baliza 1
Distancia a baliza 2

Solución:

Posición desconocida



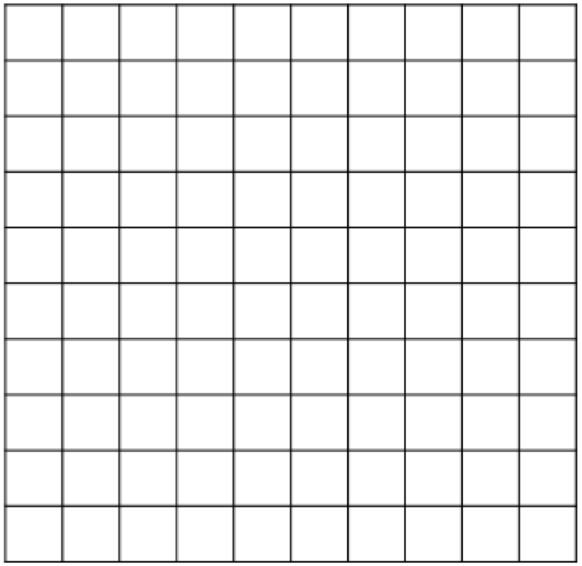
SISTEMAS NO LINEALES

Práctica:

- Encontrar soluciones de la siguiente ecuación compleja mediante el método de Newton. (N representa el número de grupo).

$$z^3 - 1/N = 0$$

- Dependiendo de la condición inicial elegida, el método tenderá a una de las tres soluciones de la ecuación.



- Discretizar el dominio [-1,1] x [-1,1] en una cuadrícula.
- Asignar un número (1,2,3) a cada solución.
- Guardar solución en archivo de texto con el formato:

<u>CI (Real)</u> \\ <u>CI (Img)</u> \\ <u>Solución (1,2,3)</u>
-1 \\ -1 \\ 2
-1 \\ -0.8 \\ 1
-1 \\ -0.6 \\ 1

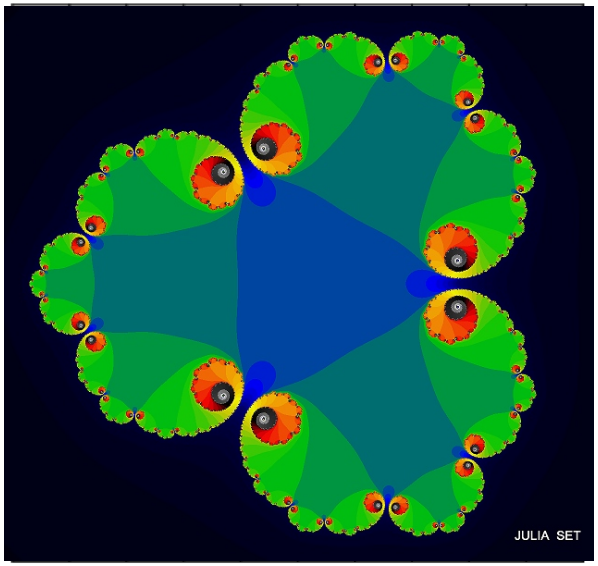
- Representar solución usando Matplotlib

Práctica:

- Encontrar soluciones de la siguiente ecuación compleja mediante el método de Newton. (N representa el número de grupo).

$$z^3 - 1/N = 0$$

- Dependiendo de la condición inicial elegida, el método tenderá a una de las tres soluciones de la ecuación.



- Discretizar el dominio [-1,1] x [-1,1] en una cuadrícula.
- Asignar un número (1,2,3) a cada solución.
- Guardar solución en archivo de texto con el formato:

```
CI (Real) \\ CI (Img) \\ Solución (1,2,3)
-1        \\ -1       \\ 2
-1        \\ -0.8     \\ 1
-1        \\ -0.6     \\ 1
```

- Representar solución usando Matplotlib